VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Určení a zobrazení vlastností selektivního systému typu dolní propust
Determinantion and Depiction of the Low Pass Filter Properties

2011 Jakub Šimůnek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Šimůnek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Určení a zobrazení vlastností selektivního systému typu dolní propust
Determinantion and Depiction of the Low Pass Filter Properties

Zásady pro vypracování:

Vhodnou technologií vytvořte výukový program sloužící k pochopení základních principů řešení obvodů pomocí admitančních modelů.
Program bude obsahovat:
1. Texty a vztahy.
2. Řešení konkrétního obvodu - interaktivní program, který pro zadané základní parametry filtru vykreslí modulovou charakteristiku obvodu.
3. Animace.
4. Implementaci závěrečného testu s vyhodnocením.

Seznam doporučené odborné literatury:

Punčochář, J.: Lineární obvody s elektronickými prvky. Skriptum, VŠB-TU Ostrava 2002
Mohylová, J.: Lineární obvody s elektronickými prvky -Sbírka příkladů, VŠB-TU Ostrava 2002
Podle pokynů vedoucího diplomové práce

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Dr. Ing. Josef Punčochář**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Dr. Ing. Josefa Punčocháře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: …..... Podpis: …....

Poděkování

Děkuji vedoucímu bakalářské práce doc. Dr. Ing. Josefu Punčochářovi za skvělý přístup a vedení mé bakalářské práce a za cenné rady při vytváření výukového programu. Dále bych poděkoval panu Ing. Janu Martinovičovi Ph.D., za pomoc s konečnou strukturou mé práce.

Abstrakt

Tato bakalářská práce se zabývá vývojem programu pro výuku problematiky filtrů typu dolní propust. První část obsahuje seznámení s problematikou frekvenčních filtrů a jejich vlastností. V druhé části se práce zabývá specifikací požadavků. Poté následuje návrh celé aplikace, ve kterém se práce zabývá samotnou strukturou programu, dále pak jednoduchým popisem ovládání. Následující část se zabývá samotnou implementací pomocí jazyka C# a popisem využitých komponent. V této kapitole jsou popsány objekty a jejich podstata v programu. Jsou zde i některé konkrétní části zdrojového kódu s následným popisem. Poslední část se zabývá postupným testováním každé části aplikace a popisem opravy chyb zjištěných při testování.

Klíčová slova

C#, dolní propust, .NET Framework, Windows Form, dynamický krok

Abstract

This bachelor´s thesis deals development of a program for teaching low-pass filters. First part involves introduction to problem and the main properties of these filters. In the other part of this work is described the propose of the application. The following section discusses the implementation program in C# language. This chapter describes the objects and other functions of this program, which was designed in C# language. This part includes some specific section of the source code with its descriptions. In the last part of this bachelor´s thesis is commented the testing of the program and the errors, which occurred and were fixed.

Key Words

C #, low-pass filter, .NET Framework, Windows Form, dynamic step

Seznam zkratek

.NET	- Platforma společnosti Microsoft
C#	- (C-Sharp) programovací jazyk
DP	- Dolní propust
GUI	- Graphical user interface (Grafické uživatelské rozhraní)
HP	- Horní propust
PP	- Pásmová propust
PZ	- Pásmová zádrž
UML	- Unified Modeling Language

Obsah

1 Úvod

Cílem této práce je implementovat výukový program pro pochopení filtrů typu dolní propust. Tato aplikace bude sloužit jako výukový program pro elektrotechnické obory. První část programu tvoří prezentace s učebními materiály. V aplikaci je dále možné simulovat modulovou charakteristiku filtrů dolní propust s různými vstupními hodnotami. Možnost vykreslení a porovnání charakteristik je užitečná funkce, jelikož programy pro vykreslování charakteristik jsou buď zdarma a nekvalitní, nebo jsou velmi drahé. Uživatelé si také mohou ověřit své znalosti v jednoduchém testu.
V bakalářské práci se nejprve věnuji stručnému popisu selektivních filtrů a požadavkům na program. V následujících kapitolách se snažím čtenáři přiblížit jednotlivé části programu. Následuje znázornění implementace celého programu a zobrazení některých důležitých kódů.

Závěr této práce je věnován zhodnocení programu a možnosti pokračování ve vývoji.

1.1 Struktura práce

Práce je rozdělena do několika kapitol.

Ve druhé kapitole si objasníme základní vlastnosti selektivních filtrů a možnosti jejich využití v praxi.

V další kapitole se zaměříme na specifikací požadavků a funkcí, které by měl program vykonávat.

Čtvrtá kapitola popisuje základní strukturu programu a v poslední řadě ovládání celé aplikace.

Následující kapitola se zabývá implementací. Popsal jsem zde strukturu systému z programátorského hlediska a využité komponenty.

Kapitola 6 popisuje, jak probíhalo testování a jaké chyby bylo potřeba opravit před konečnou verzi programu.

Poslední kapitola popisuje dosažené výsledky a možnosti dalšího vývoje.

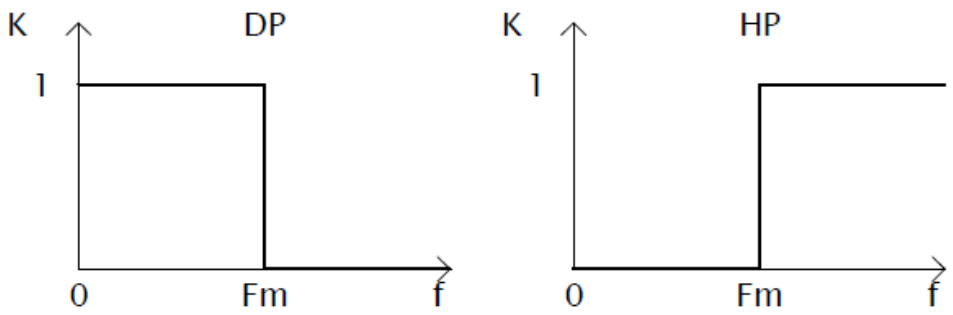
2 Selektivní filtry

Filtry jsou elektrotechnické obvody používané v mnoha oblastech elektroniky a elektrotechniky. Jejich úkolem je vybírat částí signálů v určitém frekvenčním rozsahu. Vlastnosti filtrů vyjadřujeme pomocí modulové charakteristiky. Z této charakteristiky lze vyčíst, které signály filtr propouští bez útlumu, a které s útlumem (potlačuje).

Selektivní filtry mají za úkol potlačení přenosu kmitočtových složek signálu (v nepropustném stavu), případně zesílení požadovaných složek (v propustném stavu). Filtry se využívají prakticky ve všech elektronických systémech.

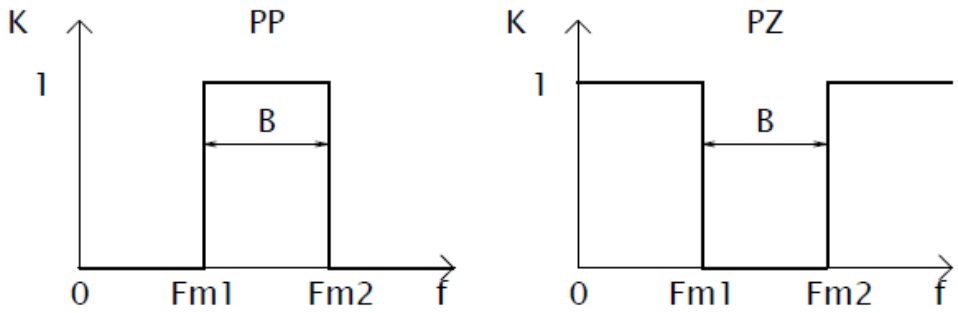
Selektivní filtry lze rozdělit podle propustného a nepropustného pásma:

- dolní propust (DP), propouští složky signálu s kmitočty nižšími než mezní kmitočet,
- horní propust (HP), propouští složky o kmitočtech vyšších než je mezní kmitočet,
- pásmová propust (PP), propouští složky signálu mezi mezním dolním a horním kmitočtem,
- pásmová zádrž (PZ), nepropouští složky signálu mezi mezním dolním a horním kmitočtem [1].

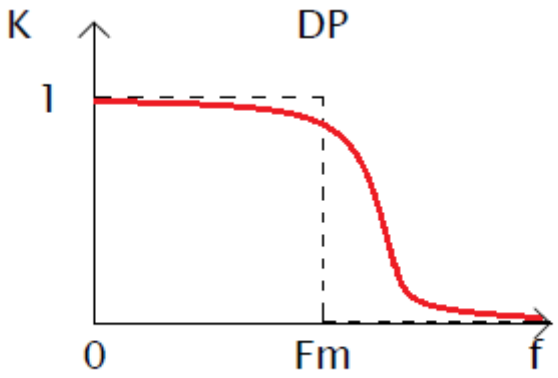


obr. 2.1: Ideální frekvenční charakteristiky pro DP a HP

Na obr. 2.1 a obr. 2.2 lze vidět ideální frekvenční charakteristiky selektivních filtrů se zidealizovaným přechodem mezi propustným a nepropustným stavem.



obr. 2.2: Ideální frekvenční charakteristiky pro PP a PZ



obr. 2.3: Reálná frekvenční charakteristika pro DP

Na obr. 2.3 je zobrazen rozdíl mezi ideální a reálnou charakteristikou. Na reálné DP (znázorněna červenou barvou) je zachycena menší strmost přechodu mezi propustným a nepropustným stavem. Aby se charakteristika reálného filtru co nejvíc přiblížila charakteristice ideálního filtru, je nutné volit co nejvyšší řády. V ideálním případě by musel být řád filtru nekonečný – takový filtr nelze realizovat.

Neideálnost filtru se projevuje u všech typů propustí.

3 Upřesnění požadavků

V této kapitole si upřesníme požadavky a funkce potřebné pro správný chod aplikace.

3.1 Upřesnění zadání

Tato aplikace je určena pro výuku filtrů typu dolní propust. Uživatel může prostudovat učební materiál k danému tématu a následně si porovnat charakteristiky filtrů s různými parametry. Součástí aplikace je i závěrečný test, ve kterém si uživatel může otestovat své znalosti.

3.2 Požadované funkce

Aplikace bude mít 3 části.

První část bude obsahovat prezentaci snímků, na kterých bude popsaná problematika filtrů typu dolní propust.

Následující část se bude zabývat vykreslováním grafů. Tato část aplikace bude požadovat následující vlastnosti:

- vykreslování grafů
- zadávání vstupních hodnot (parametrů)
- možnost výběru jednotek
- možnost vykreslení a porovnání dvou grafů
- skrytí a zobrazení hodnot vykreslených bodů

Poslední část aplikace bude tvořit kontrolní test, ve kterém si uživatelé budou testovat získané znalosti. Tato část bude obsahovat:

- náhodné načítání deseti otázek
- každá otázka bude mít 3 možné odpovědi
- automatické vyhodnocení
- uživateli se zobrazí správné i chybné odpovědi

3.3 Kdo aplikaci využije

Tuto aplikaci budou využívat studenti elektrotechnických oborů k prohlubování učiva.

4 Návrh implementace

Tato kapitola se zabývá základním popisem programu, dále pak jeho ovládáním.

4.1 Struktura programu

Program je tvořen třemi částmi. Tyto části jsou mezi sebou propojeny pomocí hlavního menu. V následujících podkapitolách si rozebereme funkce jednotlivých částí.

4.1.1 Část přednáška

Tato část je tvořena přednáškou z teorie, ve které se uživatelé seznámí s problematikou filtrům typu dolní propust. Přednáška je tvořena velkým množstvím snímků s daným tématem, kterými uživatel může jednoduše procházet.

4.1.2 Část průběh

Druhá část je podprogram pro výpočet a následné zobrazení modulové charakteristiky filtru. V této části programu si uživatel může porovnávat modulové charakteristiky při různých vstupních hodnotách.

4.1.3 Část test

Poslední část programu je tvořena kontrolním testem, který je vytvořen z deseti náhodně generovaných otázek. Každá otázka má 3 možné odpovědi, z níž je pouze jedna správná. Po zkontrolování odpovědí se může uživatel podívat na chyby.

4.2 Ovládaní programu:

Program pro ovládání využívá dvou periferií. V první části programu, která má název „přednáška“, program využívá hlavně klávesnice. V druhé a třetí části s názvy „průběh“ a „test“ se využívá především myši.

4.2.1 Ovládání části přednáška

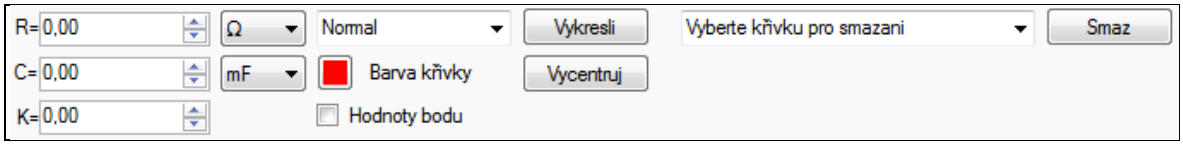
V tab. 4.1 lze vidět popis ovládání pomocí klávesnice i pomocí myši. Pokud není zapnutý mód na celou obrazovku, tak při stlačení klávesy „escape“ se přednáška vypne. Při zapnutém módu na celou obrazovku lze pomoci klávesy enter měnit snímky na následující, nebo předešlé. Směr změny je určen naposledy použitým klikem.

tab. 4.1: Ovládaní části přednáška

Ovládání přednáška			
Klávesnice		Myš	
Klávesa	Akce	Klik	Akce
šipka doleva	předešlý snímek	kliknuti na zpět	předešlý snímek
šipka doprava	další snímek	kliknuti na další	další snímek
escape	zruší mód na celou obrazovku	dvojklik	zapne/vypne mód na celou obrazovku
enter	zapne mód na celou obrazovku		

4.2.2 Ovládání části průběh

V této části programu je ovládání nastaveno hlavně pro myš. I když jde některé funkce ovládat pomocí klávesových zkratek na klávesnici, tak všechny funkce jdou ovládat i pomocí myši.



obr. 4.1: Ukázka GUI pro ovládání grafu

Na obr. 4.1 je zobrazena část GUI určená pro ovládání grafu. V levé části uživatel zadá hodnoty a vybere správné jednotky. Dále pak je třeba vybrat typ grafu, který chceme vykreslit. Jsou povoleny dva typy. Normální a logaritmický graf. Následně si vybereme barvu křivky a můžeme kliknout na tlačítko „Vykresli“, které graf vykreslí. Tlačítko „Vycentruj“ slouží k nastavení základního rozsahu os grafu. Pokud se při přibližování dostaneme někam, kde být nechceme, tak použijeme tlačítko „Vycentruj“. V poslední pravé části uživatel může vykreslené křivky, které už nepotřebuje, smazat. Ovládaní pomocí klávesnice a myši je zobrazeno v tab. 4.2 a tab. 4.3

tab. 4.2: Ovládání grafu pomocí myši

Myš - Ovládání grafu	
Klik	Akce
pohyb kolečkem dopředu a dozadu	přibližování a oddalování grafu
CTRL + Levé tlačítko myši	výběr části, kterou potřebujeme přiblížit
klik levým tlačítkem a následný pohyb	pohyb v grafu

tab. 4.3: Ovládání pomocí klávesových zkratek

Klávesnice	
Klávesa	Akce
CTRL + B	zapnutí/vypnutí hodnot bodů
CTRL + L	zobrazování legendy
CTRL + H	vypnuti/zapnuti hlavní mřížky
CTRL + M	vypnuti/zapnuti menší mřížky
CTRL + S	uloží graf jako obrázek
ALT + F4	vypne program

Pomocí myši se dále provádí:

- nastavování vstupních hodnot
- výběr jednotek daných hodnot
- nastavování barev grafu
- vybírání grafu pro smazání

4.2.3 Ovládání části test

Tato část programu je nastavená na používání jediné periferie a to myši. Uživatel zde vybere jednu z možných odpovědí a klikne na tlačítko další, nebo zpět.

5 Implementace

V této kapitole se zabývám především popisem důležitých tříd a metod využitých v programu.

5.1 Programovací jazyk

Jelikož má být aplikace přátelská k uživateli, využil jsem platformu .NET Framework, která je nejčastější používanou platformou pro operační systém Microsoft Windows. K naprogramování aplikace jsem využil jazyk C#. Pro programování v tomto jazyce jsem použil vývojové prostředí Microsoft Visual Studio 2010.

5.1.1 Základní informace o jazyku C#

C# je nový, jednoduchý, objektově orientovaný programovací jazyk vytvořený společností Microsoft. Tento jazyk je plně kompatibilní s technologií .NET a je odvozen od jazyků C, C++ a Java. Nevýhodou jazyka C# je programování časově náročné aplikace, protože jazyk nemá některé klíčové komponenty pro aplikace s velmi vysokým výkonem.

Vlastnosti Jazyka C#:

- plná podpora tříd a objektově orientovaného programování, včetně dědičnosti rozhraní i implementace, virtuálních funkcí a přetížení operátorů,
- konzistentní a vhodně definovaná sada základních typů,
- integrovaná podoba automatického generování dokumentace ve formátu XML,
- automatické čištění dynamicky přidělování paměti
- možnost označit třídy nebo metody uživatelsky definovanými atributy. Tato funkce může být užitečná pro dokumentaci a někdy má určitý vliv na překlad (např. při označení metod, aby se překládaly pouze v ladicích sestaveních),
- plný přístup ke knihovně základních tříd .NET a také snadná dostupnost rozhraní Windows API (pokud ho skutečně potřebujete, k čemuž nedochází příliš často),
- v případě potřeby jsou dostupné ukazatele a přímý přístup do paměti, ale jazyk je navržen takovým způsobem, že lze bez nich pracovat téměř ve všech situacích,
- podpora vlastností a událostí ve stylu jazyka Visual Basic,
- pouhou změnou možností překladače můžete překládat buď spustitelný soubor, nebo knihovnu komponent .NET, kterou může volat jiný kód stejným způsobem jako ovládací prvky ActiveX (komponenty COM)
- pomocí jazyka C# lze psát dynamické stránky ASP.NET a webové služby založené na XML [2]

5.2 Komponenta zedgraph

Zedgraph[3] je volně dostupná a velmi jednoduchá komponenta určená pro .NET. Pomocí této komponenty dokážeme rychle a jednoduše vytvářet různé typy grafů. U grafu lze jednoduše měnit různá nastavení. O nastavení grafu se starají dva hlavní prvky. První z prvků nese název *MasterPane*, který slučuje všechny grafy. Druhý prvek nese název *GraphPane*. V prvku jsou základní nastavení daného grafu, jako například název grafu, názvy os, typy os, minima a maxima os, nastavení hlavních i vedlejších mřížek a další. Komponenta zedgraph dokáže vykreslit různé typy 2D grafů, avšak neumožňuje vykreslování 3D grafů. Dále uvedu několik nejpoužívanějších typů grafů.

5.2.1 Některé metody a nastavení

AxisChange( ) – Velmi důležitá metoda, která nastavuje rozsah os tak, aby byly všechny body viditelné.

RestoreScale( ) – Nastaví rozsah os na defaultní hodnotu, také přiblížení vrátí na defaultní hodnotu.

SaveAsBitmap() - Otevře dialogové okno pro uložení obrázku grafu.

DoPrint( ) - Otevře dialogové okno pro tisk.

ZoomOutAll(GraphPane pane) – Zruší veškeré přiblížení a oddálí pohled na defaultní nastavení.

GraphPane.Title.Text - Nastavení názvu grafu.

GraphPane.XAxis.Title.Text – Nastavení názvu osy x.

GraphPane.YAxis.Title.Text – Nastavení názvu osy y.

GraphPane.XAxis.Type – Nastavení typu osy x. Například lineární osa, nebo logaritmická.

GraphPane.YAxis.Type - Nastavení typu osy y.

GraphPane.XAxis.MajorGrid.IsVisible – Nastavení viditelnosti hlavní mřížky osy x.

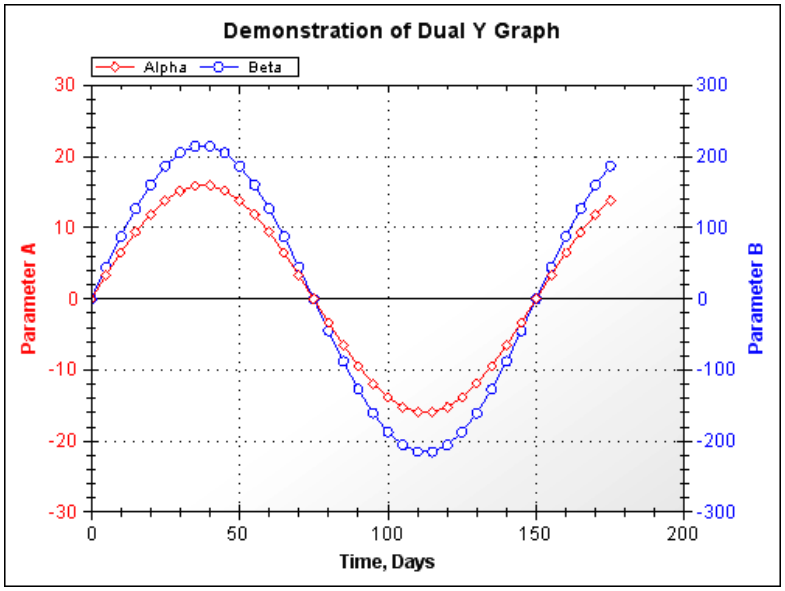
GraphPane.YAxis.MajorGrid.IsVisible – Nastavení viditelnosti hlavní mřížky osy y.

GraphPane.AddCurve()- Přidává do grafu další křivku.

ZedGraph.Refresh() - Aktualizuje graf a jeho prvky.

5.2.2 Spojnicové grafy

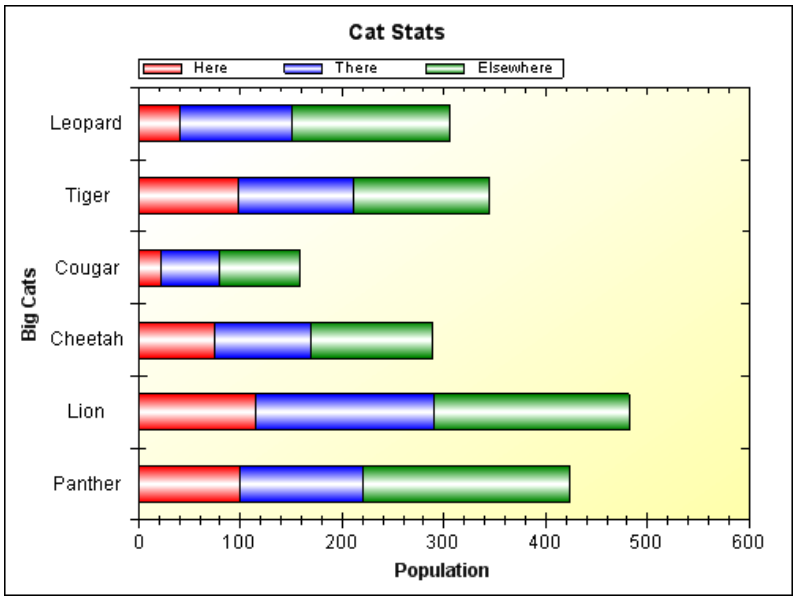
Spojnicový graf se dá definovat jako množina bodů spojená jednou čarou. Graf 5.1 ukazuje, že můžeme vykreslovat více grafů na jednom panelu.



Graf 5.1: Spojnicový graf

5.2.3 Sloupcové grafy

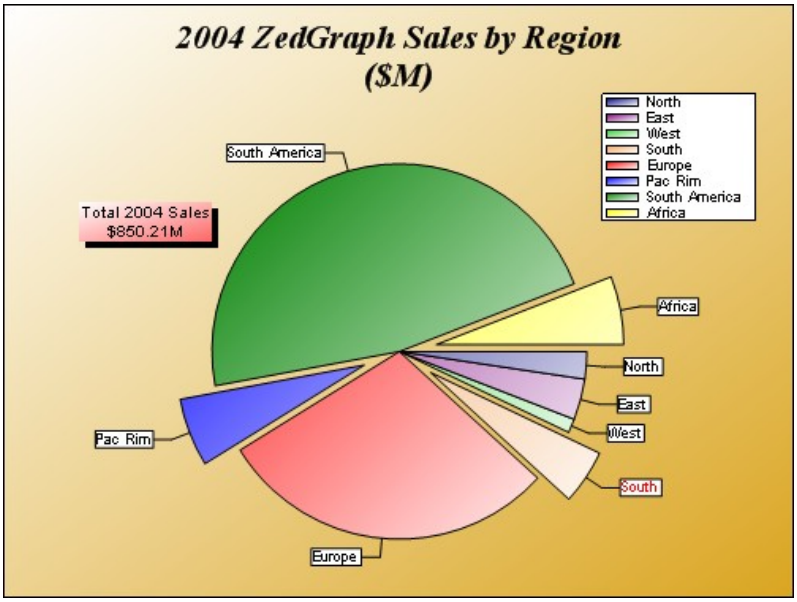
Sloupcové grafy jsou další možností komponenty zedgraph. Tyto grafy se dají vytvořit jak horizontálně, tak i vertikálně. Graf 5.2 zobrazuje ukázku horizontálního sloupcového grafu.



Graf 5.2: Sloupcový horizontální graf.

5.2.4 Výsečové grafy

Další velmi často používané grafy jsou výsečové. Nejčastěji se používají při rozdělení nějakého celku na dílčí části. Výsečové grafy nemají žádné osy. Graf 5.3 je jednoduchý výsečový graf.



Graf 5.3: Výsečový graf.

5.3 Třídní diagram

Na diagramu Diagram 5.1 je zachycen zjednodušený třídní diagram. ULM třídní diagram popisuje statickou strukturu systému pomocí tříd. Úkolem diagramu tříd je znázornit typy objektů v programu. Třída reprezentuje skupinu objektů s podobnými vlastnostmi. Objekt je instancí třídy.

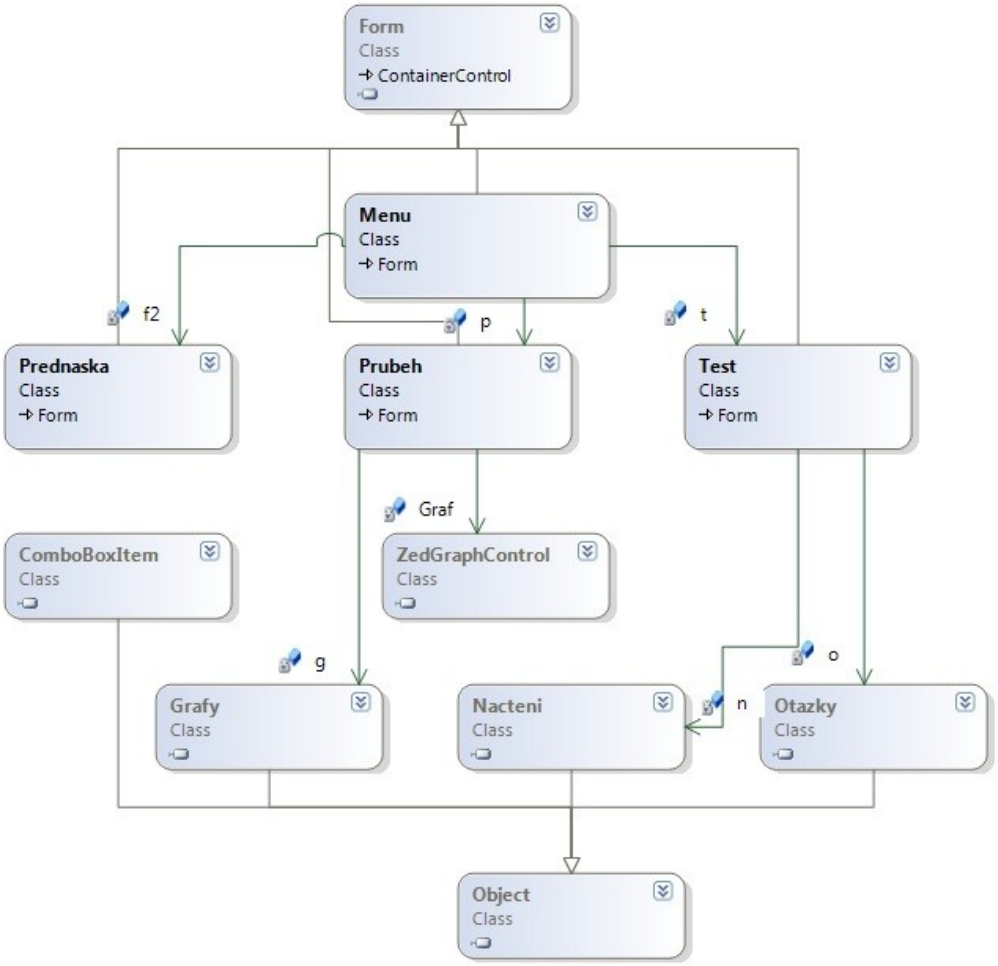
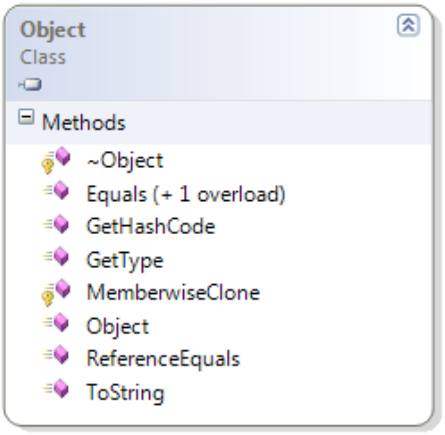


Diagram 5.1: ULM třídní diagram.

5.4 Třída Object

Třída Object je bázová třída, od které jsou odvozeny ostatní třídy. Tato třída má důležité metody pro práci s objekty. Schéma této třídy je zobrazeno na obr. 5.1. Některé metody dále popíši.



obr. 5.1: Schéma třídy Object.

Popis některých metod:
bool Equals(Object obj) – Porovnání dvou objektů.

string ToString() – Vrací řetězec, který reprezentuje aktuální objekt.

Type GetType()- Vrací instanci třídy System.Type. Tato instance má v sobě uloženou spoustu informací o aktuální třídě.

void Finalize() – Odstraňuje z paměti nevyužívaný objekt.

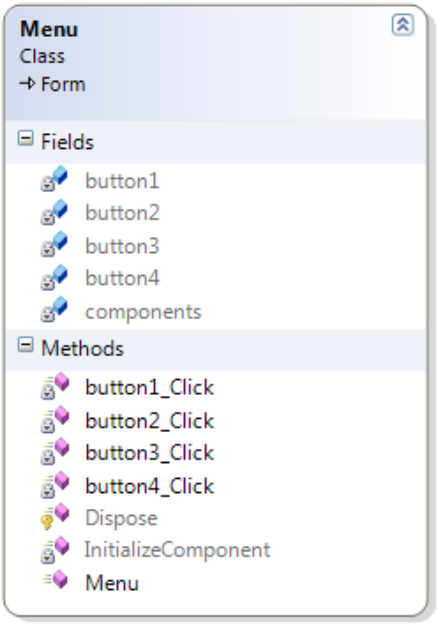
int GetHashCode() – vrací hash kód pro aktuální objekt.

Object MemberwiseClone() - vrací mělkou kopii aktuálního objektu.

bool ReferenceEquals(Object objA, Object objB) – Porovnává, zda objA odkazuje na stejnou instanci jako objB.

5.5 Windows Form Menu

Menu je první formulář, se kterým se uživatel potká. Tento formulář má úkol rozcestníku. Uživatel se zde rozhoduje, do které části programu vstoupí nejprve. V menu jsou 4 tlačítka. První tlačítko s názvem „Přednáška“ odkazuje na Windows Form Prednaska. Ve Windows Formu je prezentace k filtrům typu dolní propust. Druhé tlačítko s názvem „Grafy“ odkazuje na Windows Form Prubeh. V této části si může uživatel vykreslit charakteristiky filtrů. Třetím tlačítkem s názvem „Test“ se spouští samovyhodnocovací test. Poslední tlačítko slouží pro ukončení programu. Schéma Windows Formu Menu je znázorněno na obr. 5.2.



obr. 5.2: Schéma Windows Form Menu

button1-4_Click jsou metody prováděny po stisknutí jednotlivých tlačítek.

5.6 Windows Form Prednaska

Tato část programu je určena pro prohlížení snímků s učebním textem. V následující části podkapitoly se budeme zabývat implementací této třídy.

Nyní začneme probírat Zdrojový kód 5.1. První tři řádky tohoto kódu se zabývají nejprve zrušením okraje a poté roztáhnutím formuláře podle velikosti obrazovky. Na následujících dvou řádcích se nastavuje pozice tlačítek. Poslední řádek nastaví proměnnou „normalmod“ na false. Tuto

proměnnou program využívá pro určení, jestli je formulář v normálním módu, nebo v módu na celou obrazovku. Proměnná se využívám při ovládání aplikace.

```
this.FormBorderStyle = FormBorderStyle.None;
this.MaximumSize = new System.Drawing.Size(Screen.PrimaryScreen.Bounds.Width,
Screen.PrimaryScreen.Bounds.Height);
this.zpet.Location= new Point(odsazeni,Screen.PrimaryScreen.Bounds.Height-61);
this.dalsi.Location = new Point(Screen.PrimaryScreen.Bounds.Width - 98 -
odsazeni, Screen.PrimaryScreen.Bounds.Height - 61);
normalmod = false;
```

Zdrojový kód 5.1: Metoda fullscreen

```
this.FormBorderStyle = FormBorderStyle.Sizable;
this.MaximumSize = new System.Drawing.Size(500, 400);
this.zpet.Location = new Point(10, 292);
this.dalsi.Location = new Point(400 - 24, 400 - 108);
CenterToScreen();
normalmod = true;
```

Zdrojový kód 5.2: Tělo metody normalscreen()

Zdrojový kód 5.2 je zobrazení metody „normalscreen“. Tato metoda přepne formulář z módu na celou obrazovku na normální mód. Normální mód je okno o velikosti 500x400 pixelů. Popis kódu je obdobný jako u předešlého zdrojového kódu. Navíc je zde pouze metoda „CenterToScreen()“, která přesune formulář do středu obrazovky.

Další jednou z hlavních metod je klik na tlačítko „zpět“ a na tlačítko „další“. Zdrojový kód 5.3 zobrazuje jednu z těchto metod.

```
private void zpet_Click(object sender, EventArgs e)
{
            if (!dalsi.Enabled) dalsi.Enabled = true;
            if (snimek > 0)
            {
                snimek--;
                if (snimek==0)
                {
                    zpet.Enabled = false;
                }
                pictureBox1.Image = images[snimek];
            }
}
```

Zdrojový kód 5.3: Metoda vyvolána po klik na tlačítko zpět

Na druhém řádku zdrojového kódu je podmínka zkoumající, zda je na tlačítko „další“ povoleno kliknutí. Kliknutí na tlačítko „další“ se zakáže pouze na posledním snímku. Pokud jsme na posledním snímku a klikneme na tlačítko „zpět“, tak musíme opět povolit klikání na tlačítko „další“, abychom se mohli dostat až k poslednímu snímku. Na třetím řádku je hlavní podmínka, která zjišťuje, zda aktuální snímek má číslo větší než 0. Jelikož snímek s číslem 0 je první snímek, nesmíme programu povolit, aby došel až k hodnotě menší než 0. Na následném řádku odečítáme od proměnné „snimek“ hodnotu 1. Proměnná „snimek“ je pořadové číslo snímku, na kterém se právě nacházíme. Na pátém řádku podmínka kontroluje, zda snímek není roven nule. Je-li výsledek roven nule, tak program zakáže klikání na tlačítko „zpět“. Poslední příkaz v tomto kódu nastaví do „PictureBoxu“ snímek, který má být zobrazován a překreslí jej. Kód pro tlačítko „další“ je obdobný jako pro tlačítko „zpět“.

5.7 Windows Form Prubeh

Formulář „Prubeh“ je hlavní část programu. Uživatel si zde může vykreslit a porovnat grafy s různými hodnotami. Je zde využitá komponenta „Zedgraph“, která je pro tuto situaci vhodná viz kapitola 5.2.

```
myPane = Graf.GraphPane;
myPane.Title.Text = "Filtr";
myPane.XAxis.Title.Text = "ω";
myPane.YAxis.Title.Text = "PDP2";
myPane.XAxis.Type = AxisType.Log;
myPane.XAxis.MinorGrid.IsVisible = true;
myPane.XAxis.MajorGrid.IsVisible = true;
myPane.YAxis.MinorGrid.IsVisible = true;
myPane.YAxis.MajorGrid.IsVisible = true;
```

Zdrojový kód 5.4: Nastavení Zedgraph

Zdrojový kód 5.4 se zabývá základním nastavením komponenty „Zedgraph“. Druhý až čtvrtý řádek je nastavení popisů grafu a os. Na následném pátém řádku se určuje typ osy. Pro tento program se používá osa logaritmická. Poslední čtyři řádky nastavují viditelnost hlavní a vedlejší mřížky os.

Důležitou metodou v tomto formuláři je metoda „Vykresli“. Metoda je volána vždy, když je stlačeno tlačítko s názvem „Vykresli“. Zdrojový kód 5.5 tuto metodu zobrazuje.

Nyní se budeme zabývat implementací metody „Vykresli“. Na začátku zdrojového kódu se do proměnných r (odpor), k (zesílení) a c (kapacita) načítají uživatelem zadané hodnoty. Hodnoty se dále násobí hodnotou podle vybrané jednotky. Po načtení přichází podmínka, ve které

kontrolujeme, zda zesílení není rovno třem. Pokud je zesílení rovno třem, program uživatele informuje, aby dané zesílení změnil. Při zesílení 3 se filtr stává nestabilním. Následující podmínka kontroluje, zda jsou zadané hodnoty nenulové. Pokud jsou tyto podmínky splněny, můžeme přejít k další části metody. Vytvoříme list typu „PointPairList“, do kterého budeme později přidávat body určené k vykreslení. Následuje podmínka, ve které určujeme, jaký typ grafu se má vykreslit. Podle typu grafu se vyvolá metoda ve třídě „Grafy“, která vrátí list bodů. Tyto body uložíme do vytvořeného listu. Třídu „Grafy“ si rozebereme v další kapitole s číslem 5.8. v poslední části kódu vykreslíme křivku pomocí metody „AddCurve()“. K pomocným „pom“ a „pocitadlo“ přidám hodnotu 1. Tyto pomocné používám v legendách křivek a dále pak pro mazání křivek. Na posledním řádku metody je vyvolána metoda „AxisChange()“. Více informací o této metodě viz kapitola 5.2.1.

```
r = double.Parse(R1.Text) * Math.Pow(10,
((ComboBoxItem)Rexponenty.SelectedItem).Value);
k = double.Parse(K1.Text);
c = double.Parse(C1.Text) * Math.Pow(10,
((ComboBoxItem)Cexponenty1.SelectedItem).Value);
if (k == 3)
{
       MessageBox.Show("K se nesmí rovnat 3. Zadejte prosím jiné K");
}
else if(k==0 || r==0 || c==0)
{
       MessageBox.Show("Zadejte všechny hodnoty");
}
else
{
       list = new PointPairList();


       if (TypGrafu.SelectedIndex == 0)
       {
           list = g.NormalniGraf(r, c, k);
       }
       else
       {
           list = g.LogGraf(r, c, k);
       }
       LineItem mojeKrivka = myPane.AddCurve("Filtr "+ pom.ToString(), list,
       BarvaKrivky.Color, SymbolType.None);
       Mazani.Items.Add("Filtr " + pom.ToString());
       pom++;
       pocitadlo++;
       Graf.AxisChange();
}
```

Zdrojový kód 5.5: Tělo metody Vykresli()

Z důvodu vysokých hardwarových požadavků jsem snížil možný počet vykreslovaných křivek na 2. Díky tomuto omezení jsem přidal metodu „Smaz“, jejímž úkolem je smazání vybrané křivky z grafu. Smazání jedné křivky uvolní místo pro zakreslení nové křivky. Zdrojový kód 5.6 zobrazuje tělo metody „Smaz“.

```
if (Mazani.SelectedIndex == 0)
{
    MessageBox.Show("Musíte vybrat křivku, kterou chcete smazat.");
}
else
{
    myPane.CurveList.RemoveAt(Mazani.SelectedIndex - 1);
    Mazani.Items.RemoveAt(Mazani.SelectedIndex);
    Mazani.SelectedIndex = 0;
    Graf.AxisChange();
    Graf.Refresh();
    pocitadlo--;
}
```

Zdrojový kód 5.6: Tělo metody Smaz()

Je zde pouze jedna podmínka. Tato podmínka kontroluje, zda uživatel vybral křivku, kterou chce vymazat. Pokud je tato podmínka splněná, můžeme křivku vymazat z listu křivek (CurveList). Mazání se provádí pomocí metody „RemoveAt(int index)“. Dále musíme vymazat název křivky z ComboBoxu „Mazani“, ve kterém jsou názvy aktuálně vykreslených křivek. Jako předposlední kroky se provedou na graf metody „Refresh()“ a „AxisChange“. Informace k těmto metodám viz kapitola 5.2.1. Na posledním řádku odečteme od pomocné „pocitadlo“ číslo 1. Tato pomocná se používá při vykreslování. Určuje se podle ní počet aktuálně vykreslených křivek. Jsou li vykresleny 2 křivky, má pomocná hodnotu 2 a tudíž podmínka při vykreslování nedovolí vykreslit další křivku.

5.8 Třída Grafy

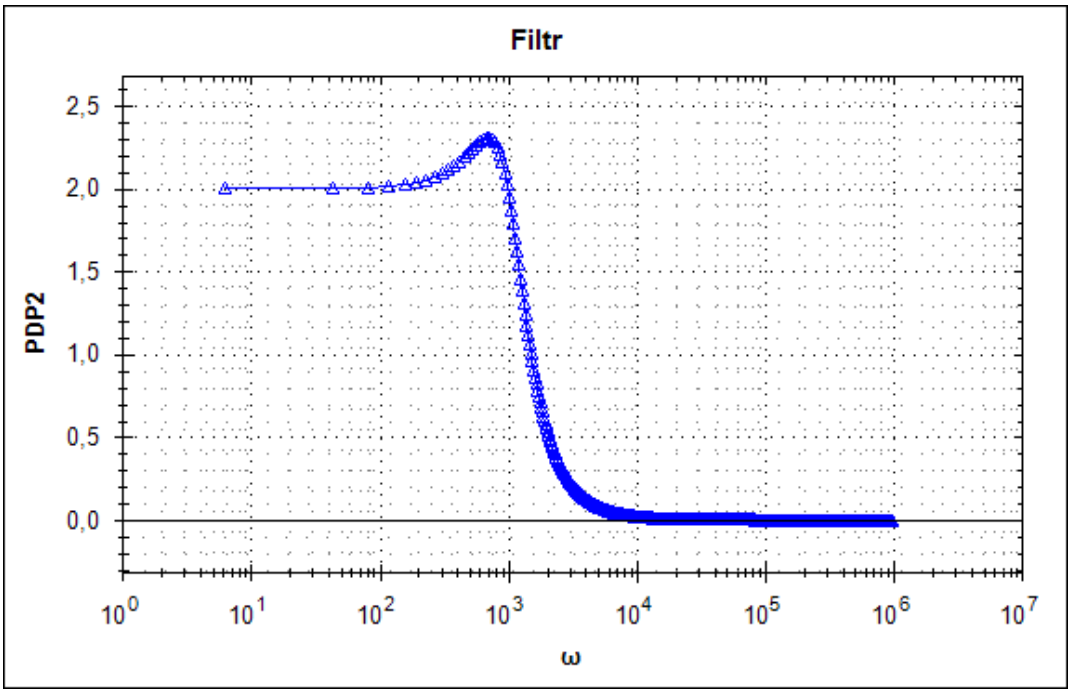
Tuto třídu jsem vytvořil pro oddělení výpočtů od zbytku kódu. Jsou zde metody s veškerými výpočty potřebnými pro vykreslení grafu. Nejdůležitějšími metodami v této třídě jsou metody „NormalniGraf“ a „LogGraf“. Tyto metody pomocí dynamického kroku vytváří list bodů, které potřebujeme k vykreslení grafu.

5.8.1 Dynamický krok

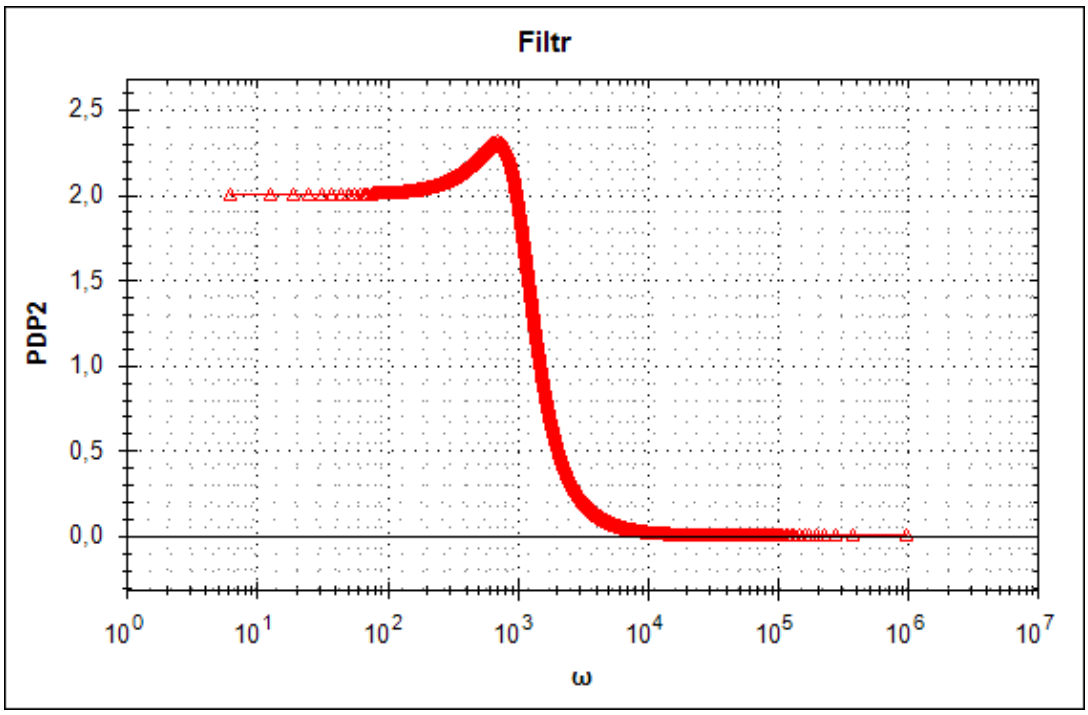
Dynamický krok je důležitou součástí programu. Díky implementaci dynamického kroku jsem urychlil výpočty a vykreslování grafu. Popis implementace dynamického kroku viz kapitola 5.8.2. Pro porovnání vykreslování pevným a dynamickým krokem jsem použil Graf 5.4 a Graf 5.5

Na grafu Graf 5.4 je znázorněno vykreslení pomocí pevného kroku. V tomto grafu lze vidět, že v pravé (konečné) části křivky zbytečně „plýtváme“ body, které bychom mohli využít na jiných částech křivky. Vykreslení tohoto grafu trvalo přibližně 24-26 ms. Největší „zásluhu“ na tak velkém čase má právě část, kdy je ω mezi hodnotami 10^4 - 10^6. Při použitém kroku 6, v této části křivky, program zakreslí přibližně 26000 bodů, což je zbytečné.

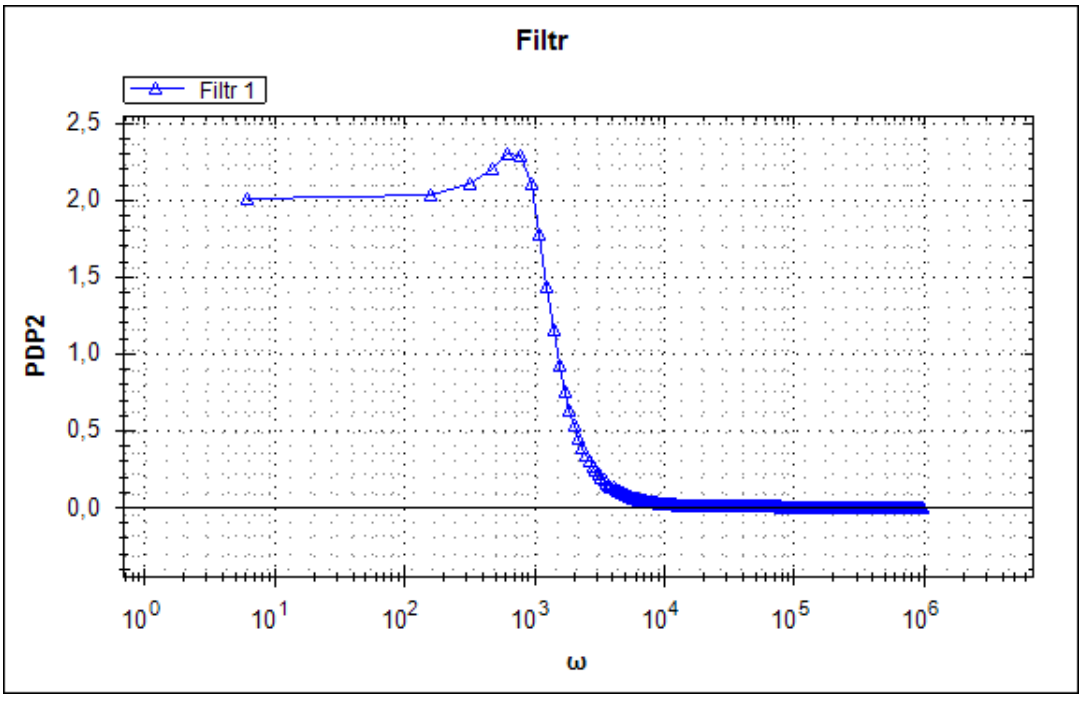
Nyní se budeme zabývat následujícím grafem. Graf 5.5 je vytvořený pro obdobné hodnoty jako předešlý graf, avšak je vykreslený pomocí dynamického kroku. Vykreslení pomocí dynamického kroku trvalo přibližně 6-7 ms, což je asi čtvrtina času při vykreslení s pevným krokem. Toto zrychlení je vytvořeno menší hustotou bodů při hodnotách ω pohybujícími se mezi 10^4 - 10^6. Uvnitř tohoto rozmezí je při testovaných hodnotách vykresleno přibližně 1500 bodů. Na grafu jde dále vidět větší hustota bodů v prostření části. Tato větší hustota bodu nám vytváří přesnější křivku.



Graf 5.4: Vykreslení pomocí pevného kroku



Graf 5.5: Vykreslení pomocí dynamického kroku



Graf 5.6: Vykreslení pomocí dynamického kroku při čase 6-7 ms

Graf 5.6 je vykreslen pomocí pevného kroku. Tento graf je vytvořen pro stejné hodnoty jako předešlé dva grafy. Při vykreslení této křivky jsem použil krok 25, což je téměř 4x větší krok než na grafu Graf 5.4. Krok 25 jsem použil, abych snížil čas vykreslování na 6-7 ms (čas obdobný vykreslování pomocí dynamického kroku). Při porovnání Graf 5.5 a Graf 5.6 lze vidět, jakou výhodu přináší vykreslování pomocí dynamického kroku.

5.8.2 Výpočet dynamického kroku

Rozdíl mezi dynamickým a pevným krokem je popsán v kapitole 5.8.1.

```
w0 = W0(r, c);
q = Q(k);
w2 = W2(r, c);
f = 1;
f_max = (long)Math.Round((w0 * 100) / 2 * Math.PI);
max = false;
body = new PointPairList();
if(w0 > Math.Pow(10, 12)){
       krok_max = 20076800000;
       snizit = 0.009;
       zvysit = 0.0009;
}
else if ((w0 < Math.Pow(10, 12) &&(w0 >= Math.Pow(10, 9)))){
       krok_max = 2507680000;
       snizit = 0.008;
       zvysit = 0.0008;
}
else if ((w0 < Math.Pow(10, 9) && (w0 >= Math.Pow(10, 6)))){
       krok_max = 20388608;
       snizit = 0.005;
       zvysit = 0.0005;
}
else if ((w0 < Math.Pow(10, 6) && (w0 >= Math.Pow(10, 3)))){
       krok_max = 262144;
       snizit = 0.0001;
       zvysit = 0.00001;
}
else if ((w0 < Math.Pow(10, 3) && (w0 >= Math.Pow(10, 2)))){
       krok_max = 65536;
       snizit = 0.0001;
       zvysit = 0.000001;
}
else if (w0 < Math.Pow(10, 2)){
       krok_max = 16384;
       snizit = 0.00001;
       zvysit = 0.000001;
}
```

Zdrojový kód 5.7: první část metody NormalniGraf

Nyní si rozebereme první část kódu zabývající se implementací dynamického kroku. Zdrojový kód 5.7 zobrazuje tuto metodu. Nejprve vypočítáme některé potřebné hodnoty. První hodnota „w0“ je úhlová rychlost při mezní frekvenci. Mezní frekvence je kmitočet při poklesu o 3 dB od maximální hodnoty. Jednoduše řečeno je to frekvence v místě, kde přibližně charakteristika filtru začíná klesat. V další části pak výpočet „q“ (činitel jakosti). Následně ještě spočítám „w2“, které je druhou mocninou“w0“. Jako poslední nastavíme rozsah frekvencí, pro které budeme následné body počítat. Nyní jsme se v kódu dostali k několika podmínkám. Tyto podmínky zjišťují, jaký rozsah bude potřeba pro zobrazení grafu. Nastavují se zde 3 hodnoty. První hodnota s názvem „krok_max“ je nastavení maximální velikosti kroku. Maximální hodnotu kroku zde definuji, aby byl graf přesnější. Hodnoty „snizit“ a „zvysit“ slouží k počítání kroku.

Zdrojový kód 5.8 znázorňuje smyčku, která počítá hodnoty vykreslovaných bodů. Rozebereme si daný kód. Na prvním řádku začíná smyčka. Tato smyčka se ukončí, jakmile hodnota frekvence „f“ dosáhne hodnoty f_max. Ve smyčce se nejprve z frekvence „f“ vypočte uhlová rychlost „w“. Tato uhlová rychlost je x-ová souřadnice bodu. Následně vypočteme modul přenosu, který zapíšeme do proměnné „novy“. Následující podmínka zjišťuje, zda je frekvence „f“ menší, nebo rovna 10. Pokud je tato podmínka splněna následuje zapsání hodnot do listu bodů, poté do proměnné „stary“ nastavíme hodnotu „novy“. Nakonec se k frekvenci přičte krok a cyklus pokračuje. Tato podmínka je zde proto, aby prvních pár hodnot bylo zapsáno s klasickým krokem. Pokud podmínka není splněná (to znamená, že frekvence je větší než 10), tak smyčka přechází do další části. V této části následuje další podmínka, ve které počítáme rozdíl hodnot „stary“ a „novy“. Pokud je absolutní hodnota rozdílu těchto hodnot větší než hodnota „snizit“ a krok lze zmenšit, tak odečteme od „f“ hodnotu „ krok“, poté snížíme krok na polovinu, nastavíme proměnnou „stav“ na false, přidáme k „f“ nově zmenšenou hodnotu „krok“ a program nás vrátí na začátek smyčky. Jednoduše řečeno, pokud je velký rozdíl mezi sousedními body, zkrátí se krok a poslední hodnota se počítá znova s menším krokem. Pokud tato podmínka není splněná, posouváme se v cyklu dále. Zde se nalézá podobná podmínka. Počítáme opět absolutní hodnotu rozdílu proměnných „novy“ a „stary“. Pokud je tato hodnota menší než pomocná proměnná „zvýšit“ a zároveň má proměnná „max“ hodnotu false, je tato podmínka splněna a následuje kontrola, zda proměnná „krok“ nepřekračuje maximální hodnotu kroku. Následná podmínka sleduje, zda je proměnná „stav“ nastavená na hodnotu true. Stav má hodnotu false pokud byla v minulem cyklu snížena proměnná „krok“. Tato pomocná proměnná je zde, aby se cyklus nezacyklil. Pokud by v kódu tato proměnná nebyla, mohlo by dojít ke snížení kroku, poté opět k jeho zvýšení a tento sled událostí by se pořád opakoval. Dostali jsme se až k poslední podmínce, která kontroluje, zda frekvence „f“ + „krok“ nepřesahuje hodnotu maximální frekvence. Pokud „f“ + „krok“ přesahuje hodnotu „f_max“, nastaví se proměnná „max“ na hodnotu true a cyklus se provede naposledy. Pokud ovšem tato podmínka není splněna, zvýší se „krok“ na dvojnásobek a cyklus pokračuje. Jestliže se krok nemění, prochází cyklus až ke konci a zde se přidá bod do listu bodů a následně se do hodnoty

„stary“ nastaví hodnota novy“, proměnné „stav“ přiřadíme hodnotu true a nakonec se k „f“ přičte „krok“. Smyčka pokračuje, dokud je „f“ menší než „f_max“.

```
while (f <= f_max){
    w = W(f);
    novy = Vypocet(k, w2, q);
    if (f <= 10){
           body.Add(w, novy);
           stary = novy;
           f += krok;
    }
    else{
           if (Math.Abs(novy - stary) > snizit){
                  if (krok / 2 >= 0.5){
                  f -= krok;
                  krok /= 2;
                  stav = false;
                  f += krok;
                  continue;
                  }
           }
           else if ((Math.Abs(novy - stary) < zvysit) && (max == false)){
                  if (krok <= krok_max){
                        if (stav){
                               if ((f + (krok * 2)) > f_max){
                                      max = true;
                                      f += krok;
                                      continue;
                               }
                               else{
                                      f -= krok;
                                      krok *= 2;
                                      f += krok;
                                      continue;
                               }
                        }
                  }
           }
           stary = novy;
           body.Add(w, novy);
           stav = true;
           f += krok;
     }
} return body;
```

Zdrojový kód 5.8: Druhá část metody NormalniGraf

5.9 Windows Form Test

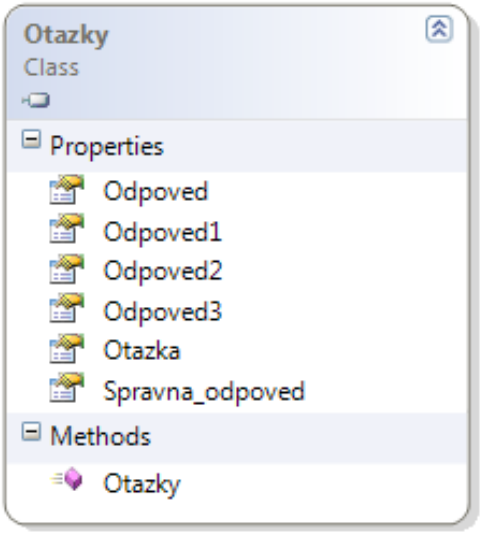
Tento Windows Form tvoří třetí část programu, kterou je samovyhodnocovací test. Uživatel vyplňuje náhodně generovaný test, který obsahuje 10 otázek. Každá otázka má 3 možné odpovědi, z toho je pouze jedna správná. Po zodpovězení poslední otázky se test vyhodnotí a uživatel se může podívat, kde udělal chybu.

5.10 Třída Nactení

Tuto třídu využívá Windows Form Test. Více informací o třídě „Test“ viz kapitola 5.9. Třída „Nacteni“ obsahuje metodu, která načítá otázky z binárního souboru. Otázky se pomocí metody „Nacti()“ načtou do listu, který je na konci metody předán třídě „Test“.

5.11 Třída Otazky

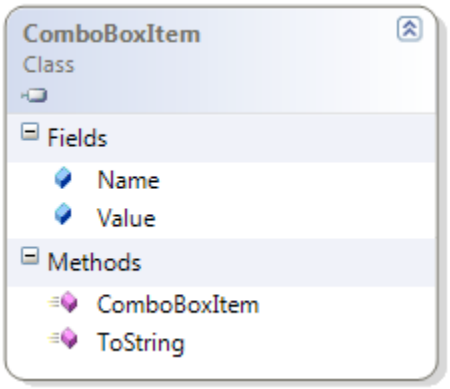
Třída „Otazky“ obsahuje veškeré informace o otázce. Tato třída je pomocnou třídou při vytváření otázek. Windows Form Test vytváří pro každou otázku jednu instanci této třídy. Pomoci těchto instancí se program jednoduše pohybuje mezi otázkami. Schéma třídy je zobrazeno na obr. 5.3.



obr. 5.3: Schéma třídy Otazky

5.12 Třída ComboBoxItem

Třída „ComboBoxItem“ je pomocnou třídou při zadávání hodnot pro vykreslení. Pomocí této třídy uživatel může při zadávání hodnot vybírat přímo jednotky. Instance této třídy je objekt obsahující dvě proměnné. Tyto objekty se vpisují do „ComboBoxu“ v části Windows Form Prubeh. Do proměnné „název“ zapisujeme značku jednotky (např. nF) a do proměnné „Value“ přiděluji hodnotu exponentu k základní jednotce. Jestliže nF $= 10^{-9}$F, tak program do proměnné „Value“ zapíše hodnotu -9. Schéma třídy „ComboBoxItem“ je znázorněno na obr. 5.4.



obr. 5.4: Schéma třídy ComboBoxItem

5.13 Třída VlozOtazky

Tuto třídu jsem vytvořil pouze pro jedno použití, a to pro vytvoření binárního souboru s otázkami. Vytvářím zde instance třídy „Otázka“, do kterých vkládám všechny potřebné informace o dané otázce a jejich odpovědích. Tyto instance ukládám do listu. Nakonec celý list otázek zapíši do binárního souboru „Test.bin“.

6 Testování

V této kapitole se budu zabývat testováním programu a následným odstraňováním chyb. Testování je důležitou funkcí při programování jakékoliv aplikace.

6.1 Testování části přednáška

Jelikož se jedná o obrázkovou prezentaci, jednou z nejdůležitějších částí pro testování bylo zobrazení. Zprvu aplikace vypadala velmi dobře. Začal jsem tedy testovat aplikaci na počítačích s jiným rozlišením obrazovky a na starších systémech Windows. Při testech jsem narazil na řadu chyb v zobrazení této části programu na starších systémech. Na základě poznatků z testů jsem přestavěl GUI tak, aby vyhovovalo i starším systémům.

Po úpravě aplikace jsem začal opět s testy. Tentokrát jsem již na žádný problém nenarazil

6.2 Testování části průběh

V této části se zobrazením nebyl problém. Testování jsem zaměřoval na část s výpočty. Při prvním testování jsem narazil na problémy s pevným krokem. Nastávaly celkem dva problémy. První problém nastal při vykreslování vyšších frekvencí. Zde jsem musel na výsledný graf čekat i několik sekund. Druhý problém byl s nepřesností grafu. Body byly v důležitých oblastech velmi vzdálené, což znamená, že graf byl skreslený. Oba tyto problémy jsem následně vyřešil pomocí dynamického kroku popsaného v kapitole 5.8.1. Implementace dynamického kroku je popsána v kapitole 5.8.2.

Jakmile jsem do aplikace přidal dynamický krok, opět začalo testování. V tomto testování byla aplikace o mnoho rychlejší. Nicméně i tentokrát jsem našel velmi podobné chyby, i když v daleko menším měřítku. Tyto chyby jsem opravil pomocí nastavení citlivostí. Napsal jsem část kódu, která pomocí vstupních hodnot vypočítá, v jakých hodnotách bude graf vykreslen a podle těchto výpočtů nastavuji citlivost a maximální délku dynamického kroku.

Začal jsem tedy další testování, ve kterém jsem už neobjevil žádné další chyby.

6.3 Testovaní části test

Při testování této části jsem po krátké době našel chybu, při které chtěl program z pole získat neexistující hodnotu. Tato chyba byla způsobena chybně nastavenou proměnnou. Následovalo další testování, ve kterém již bylo vše v pořádku.

7 Závěr

Cílem bakalářské práce bylo navrhnout a implementovat výukový program s testem znalostí na téma filtry typu dolní propust.

Bakalářská práce nejprve popisuje základní informace o selektivních filtrech. Následující část ukazuje návrh a rozdělení programu. Poté nastupuje samostatné programové řešení. Závěr práce popisuje testování programu a následné opravy.

Program je implementován v jazyce C# s podporou technologie .NET Framework 4.0. Aplikace je rozdělena do tří podprogramů tvořící jeden celek. Důležitou součástí programu je vykreslování modulové charakteristiky, které se pro pochopení látky velmi hodí.

Program jsem testoval na systémech Windows XP a Windows 7. Pokud je v systémech nainstalovaný .NET Framework 4.0, není problém s plnou funkčností programu.

Všechny požadavky zadání byly splněny.

Z práce jsem pochytil mnoho zkušeností s programováním v C#, které v budoucnu mohu uplatnit v praxi.

7.1 Možný vývoj

Možností pro rozšíření programu je mnoho. Mimo přidávání dalších témat lze program rozšiřovat o vykreslování různých typů filtrů, případně i o jiné typy charakteristik.

8 Použitá literatura

[1] HÁJEK, Karel, SEDLÁČEK, Jiří. *Kmitočtové filtry*. 1. vyd. Praha : BEN, 2002. 536 s. ISBN 80-7300-023-7.

[2] NAGEL, Christian, et al. *C# 2008 : Programujeme profesionálně*. Vyd. 1. Brno : Computer Press, 2009. 1126 s. ISBN 978-80-251-2401-7.

[3] *SourceForge* [online]. 2008-12-12 [cit. 2011-04-29]. ZedGraph. Dostupné z WWW: <http://sourceforge.net/projects/zedgraph/>.

[4] PUNČOCHÁŘ, Josef. *Operační zesilovače v elektronice*. 5. vyd. Praha : BEN, 2002. 495 s. ISBN 80-7300-059-8.

9 Přílohy

Součásti originálu bakalářské práce je CDROM obsahující tuto práci a veškeré zdrojové kódy.

9.1 Obsah přiloženého CD

- V adresáři program je pouze program
- V adresáři Zdrojový kód je program se všemi zdrojovými kódy
- V adresáři bakalářská práce je tato práce v elektronické podobě